Chapter 25

DIGITAL COMPUTERS APPLIED TO GAMES

Chess problems are the hymn tunes of mathematics—G. H. HARDY

MACHINES WHICH WILL PLAY GAMES have a long and interesting history. Among the first and most famous was the chess-playing automaton constructed in 1769 by the Baron Kempelen; M. Maelzel took it on tour all over the world, deceiving thousands of people into thinking that it played the game automatically. This machine was described in detail by Edgar Allan Poe; it is said to have defeated Napoleon himself—and he was accounted quite a good player, but it was finally shown up when somebody shouted "FIRE" during a game, and caused the machine to go into a paroxysm owing to the efforts of the little man inside to escape.

In about 1890 Signor Torres Quevedo made a simple machine—a real machine this time—which with a rook and king can checkmate an opponent with a single king. This machine avoids stalemate very cleverly and always wins its games. It allows an opponent to make two mistakes before it refuses to play further with him, so it is always possible to cheat by moving one's own king the length of the board. The mechanism of the machine is such that it cannot move its rook back past its king and one can then force a draw! This machine, like Babbage's "noughts and crosses" machine is relatively simple, the rules to be obeyed are quite straightforward, and the machines couldn't lose. Babbage thought that his analytical engine ought to be able to play a real game of chess, which is a much more difficult thing to do.

In this chapter we discuss how a digital computer can be made to play chess—it does so rather badly, and how it plays draughts—it does so quite well. We shall also describe a special simple machine which was built to entertain the public during the Festival of Britain. It was called Nimrod because it played nim, a game which is like noughts and crosses, in that the tricks which are needed to win can be expressed in mathematical terms. This machine was on show in South Kensington for six months and took on all comers.

During the Festival the Society for Psychical Research came and fitted up a room nearby in order to see if the operations of the machine could be influenced by concentrated thought on the part of the research workers, most of whom were elderly ladies. When this experiment had failed they tried to discover whether they in turn could be affected by vibrations from the machine, and could tell from another room how the game was progressing. Unfortunately this experiment, like the first, was a complete failure, the only conclusion being that machines are much less co-operative than human beings in telepathic experiments.

At the end of the Festival of Britain Nimrod was flown to Berlin and shown at the Trade Fair. The Germans had never seen anything like it, and came to see it in their thousands, so much so in fact that on the first day of the show they entirely ignored a bar at the far end of the room where free drinks were available, and it was necessary to call out special police to control the crowds. The machine became even more popular after it had defeated the Economics Minister, Dr. Erhardt, in three straight games. After this it was taken to Canada and demonstrated to the Society of Engineers in Toronto. It is still somewhere on the North American continent, though it may have been dismantled by now, and it would be amusing to match it against some of the other nim-playing machines which have been built in the last year or two.

The reader might well ask why we bother to use these complicated and expensive machines in so trivial a pursuit as playing games. It would be disingenuous of us to disguise the fact that the principal motive which prompted the work was the sheer fun of the thing, but nevertheless if ever we had to justify the time and effort (and we feel strongly that no excuses are either necessary or called for) we could quite easily make a pretence at doing so. We have already explained how hard all programming is to do, and how much difficulty is due to the incompetence of the machine at taking an overall view of the problem which it is analysing. This particular point is brought out more clearly in playing games than in anything else. The machine cannot look at the whole of a chess board at once; it has to peer short-sightedly at every square in turn, in much the same way as it has to look at a commercial document. Research into the techniques of programming a machine to tackle complicated problems of this type may in fact lead to quite important advances, and help in serious work in business and economics—perhaps, regrettably, even in the theory of war. We hope that mathematicians will continue to play draughts and chess, and to enjoy themselves as long as they can.

We have often been asked why we don't use the machine to work out the football pools, or even to do something to remove the present uncertainty about the results of tomorrow's horse races. Perhaps one day we shall persuade our mathematicians to apply themselves to this problem too. It would first be necessary to establish a series of numerical criteria from which the machine could predict the results with greater certainty than the ordinary citizen can achieve with his pin; the presumption underlying the whole idea is that such criteria do in fact exist, but that they are too complicated for a man to apply in time, whereas a machine could do the necessary computations for him. It is most unlikely that a machine could ever hope to predict (for example) the results of a single football match, but it is at least possible that a detailed analysis might significantly improve the odds in favour of the gambler, so that if he invested on a large enough scale he could make a profit. It is notoriously true that mathematics, and particularly the theory of probability, owes more to gambling than gambling owes to mathematics; perhaps a machine might do something to restore the balance. Lady Lovelace lost a fortune by trying to back horses scientifically, and many others have done the same; all one could hope for is a slight improvement in the odds. We might make it pay but we doubt it; as an academic exercise it would be amusing, but we shall give the project a low priority.

CHESS

When one is asked, "Could one make a machine to play chess?" there are several possible meanings which might be given to the words. Here are a few—

(*a*) Could one make a machine which would obey the rules of chess, i.e. one which would play random legal moves, or which could tell one whether a given move is a legal one?

(*b*) Could one make a machine which would solve chess problems, e.g. tell one whether, in a given position, white has a forced mate in three?

(*c*) Could one make a machine which would play a reasonably good game of chess, i.e. which, confronted with an ordinary (that is, not particularly unusual) chess position, would after two or three minutes of calculation, indicate a passably good legal move?

(*d*) Could one make a machine to play chess, and to improve its play, game by game, profiting from its experience?

To these we may add two further questions, unconnected with chess, which are likely to be on the tip of the reader's tongue.

(*e*) Could one make a machine which would answer questions put to it, in such a way that it would not be possible to distinguish its answers from those of a man?

(*f*) Could one make a machine which would have feelings as you and I have?

The problem to be considered here is (*c*), but to put this problem into perspective with the others I shall give the very briefest of answers to each of them.

To (*a*) and (*b*) I should say, "This certainly can be done. If it has not been done already it is merely because there is something better to do."

Question (*c*) we are to consider in greater detail, but the short answer is, "Yes, but the better the standard of play required, the more complex will the machine be, and the more ingenious perhaps the designer."

To (*d*) and (*e*) I should answer, "I believe so. I know of no really convincing argument to support this belief, and certainly of none to disprove it."

To (*f*) I should say, "I shall never know, any more than I shall ever be quite certain that *you* feel as I do."

In each of these problems except possibly the last, the phrase, "Could one make a machine to . . ." might equally well be replaced by, "Could one programme an electronic computer to . . ." Clearly the electronic computer so programmed would itself constitute a machine. And on the other hand if some other machine had been constructed to do the job we could use an electronic computer (of sufficient storage capacity), suitably programmed, to calculate what this machine would do, and in particular what answer it would give.

After these preliminaries let us give our minds to the problem of making a machine, or of programming a computer, to play a tolerable game of chess. In this short discussion it is of course out of the question to provide actual programmes, but this does not really matter on account of the following principle—

If one can explain quite unambiguously in English, with the aid of mathematical symbols if required, how a calculation is to be done, then it is always possible to programme any digital computer to do that calculation, provided the storage capacity is adequate.

This is not the sort of thing that admits of clear-cut proof, but amongst workers in the field it is regarded as being clear as day. Accepting this principle, our problem is reduced to explaining "unambiguously in English" the rules by which the machine is to choose its move in each position. For definiteness we will suppose the machine is playing white.

If the machine could calculate at an infinite speed, and also had unlimited storage capacity, a comparatively simple rule would suffice, and would give a result that in a sense could not be improved on. This rule could be stated:

"Consider every possible continuation of the game from the given position. There is only a finite number of them (at any rate if the fifty-move rule makes a draw obligatory, not merely permissive). Work back from the end of these continuations, marking a position with white to play as 'win' if there is a move which turns it into a position previously marked as 'win.' If this does not occur, but there is a move which leads to a position marked 'draw,' then mark the position 'draw.' Failing this, mark it 'lose.' Mark a position with black to play by a similar rule with 'win' and 'lose' interchanged. If after this process has been completed it is found that there are moves which lead to a position marked 'win,' one of these should be chosen. If there is none marked 'win' choose one marked 'draw' if such exists. If all moves lead to a position marked 'lose,' any move may be chosen."

Such a rule is practically applicable in the game of noughts and crosses, but in chess is of merely academic interest. Even when the rule can be applied it is not very appropriate for use against a weak opponent, who may make mistakes which ought to be exploited.

In spite of the impracticability of this rule it bears some resemblance to what one really does when playing chess. One does not follow all the continuations of play, but one follows some of them. One does not follow them until the end of the game, but one follows them a move or two, perhaps more. Eventually a position seems, rightly or wrongly, too bad to be worth further consideration, or (less frequently) too good to hesitate longer over. The further a position is from the one on the board the less likely it is to occur, and therefore the shorter is the time which can be assigned for its consideration. Following this idea we might have a rule something like this—

"Consider all continuations of the game consisting of a move by white, a reply by black, and another move and reply. The value of the position at the end of each of these sequences of moves is estimated

according to some suitable rule. The values at earlier positions are then calculated by working backwards move by move as in the theoretical rule given before. The move to be chosen is that which leads to the position with the greatest value."

It is possible to arrange that no two positions have the same value. The rule is then unambiguous. A very simple form of values, but one not having this property, is an "evaluation of material," e.g. on the basis—

$$\begin{aligned} \text{P} &= 1 \\ \text{Kt} &= 3 \\ \text{B} &= 3\tfrac{1}{2} \\ \text{R} &= 5 \\ \text{Q} &= 10 \\ \text{Checkmate} &= 1000 \end{aligned}$$

If B is black's total and W is white's, then W/B is quite a good measure of value. This is better than $W - B$ as the latter does not encourage exchanges when one has the advantage. Some small extra arbitrary function of position may be added to ensure definiteness in the result.

The weakness of this rule is that it follows all combinations equally far. It would be much better if the more profitable moves were considered in greater detail than the less. It would also be desirable to take into account more than mere "value of material."

After this introduction I shall describe a particular set of rules, which could without difficulty be made into a machine programme. It is understood that the machine is white and that white is next to play. The current position is called the *position on the board*, and the positions arising from it by later moves *positions in the analysis*.

"Considerable" Moves

"Considerable" here is taken to mean moves which will be "considered" in the analysis by the machine.

Every possibility for white's next move and for black's reply is "considerable." If a capture is considerable then any recapture is considerable. The capture of an undefended piece or the capture of a piece of higher value by one of lower value is always considerable. A move giving checkmate is considerable.

Dead Position

A position in the analysis is dead if there are no considerable moves in that position, i.e. if it is more than two moves ahead of the present position, and no capture or recapture or mate can be made in the next move.

Value of Position

The value of a dead position is obtained by adding up the piece values as above, and forming the ratio *W/B* of white's total to black's. In other positions with white to play the value is the greatest value of (*a*) the positions obtained by considerable moves, or (*b*) the position itself evaluated as if a dead position. The latter alternative is to be omitted if all moves are considerable. The same process is to be undertaken for one of black's moves, but the machine will then choose the *least* value.

Position-play Value

Each white piece has a certain position-play contribution and so has the black king. These must all be added up to give the position-play value.

For a Q, R, B, or Kt, count—

(*a*) The square root of the number of moves the piece can make from the position, counting a capture as two moves, and not forgetting that the king must not be left in check.

(*b*) (If not a Q) 1·0 if it is defended, and an additional 0·5 if twice defended.

For a K, count—

(*c*) For moves other than castling as (*a*) above.

(*d*) It is then necessary to make some allowance for the vulnerability of the K. This can be done by assuming it to be replaced by a friendly Q on the same square, estimating as in (*a*), but subtracting instead of adding.

(*e*) Count 1·0 for the possibility of castling later not being lost by moves of K or rooks, a further 1·0 if castling could take place on the next move, and yet another 1·0 for the actual performance of castling.

For a P, count—

(*f*) 0·2 for each rank advanced.

(*g*) 0·3 for being defended by at least one piece (not P).

For the black K, count—

(*h*) 1·0 for the threat of checkmate.

(*i*) 0·5 for check.

We can now state the rule for play as follows. The move chosen must have the greatest possible value, and, consistent with this, the greatest possible position-play value. If this condition admits of several solutions a choice may be made at random, or according to an arbitrary additional condition.

Note that no "analysis" is involved in position-play evaluation. This is to reduce the amount of work done on deciding the move.

The game below was played between this machine and a weak player who did not know the system. To simplify the calculations the square roots were rounded off to one decimal place, i.e. this table was used—

Number . . .	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Square Root . .	0	1	1·4	1·7	2·0	2·2	2·4	2·6	2·8	3·0	3·2	3·3	3·5	3·6

No random choices actually arose in this game. The increase of position-play value is given after white's move if relevant. An asterisk indicates that every other move had a lower position-play value.

	White (*Machine*)		*Black*
1.	P—K 4	4·2*	P—K 4
2.	Kt—Q B 3	3·1*	Kt—K B 3
3.	P—Q 4	2·6*	B—Q Kt 5
4.	Kt—K B 3(1)	2·0	P—Q 3
5.	B—Q 2	3·5*	Kt—Q B 3
6.	P—Q 5	0·2	Kt—Q 5
7.	P—K R 4(2)	1·1*	B—Kt 5
8.	P—Q R 4(2)	1·0*	Kt×Kt ch.
9.	P×Kt		B—K R 4
10.	B—Kt 5 ch.	2·4*	P—Q B 3
11.	P×P		O—O
12.	P×P		R—Kt 1
13.	B—R 6	−1·5	Q—R 4
14.	Q—K 2	0·6	Kt—Q 2
15.	K R—Kt 1(3)	1·2*	Kt—B 4(4)
16.	R—Kt 5(5)		B—Kt 3
17.	B—Kt 5	0·4	Kt×Kt P
18.	O—O—O	3·2*	Kt—B 4
19.	B—B 6		K R—Q B 1
20.	B—Q 5		B×Kt
21.	B×B	0·7	Q×P
22.	K—Q 2		Kt—K 3
23.	R—Kt 4	−0·3	Kt—Q 5
24.	Q—Q 3		Kt—Kt 4
25.	B—Kt 3		Q—R 3
26.	B—B 4		B—R 4
27.	R—Kt 3		Q—R 5
28.	B×Kt		Q×B
29.	Q×P (6)		R—Q 1(4)
30.	Resigns(7)		

Notes—

1. If B—Q 2 3·7* then P×P is foreseen.
2. Most inappropriate moves.
3. If white castles then B×Kt, B×B, Q×P.
4. The fork is unforeseen at white's last move.
5. Heads in the sand!
6. Fiddling while Rome burns!
7. On the advice of his trainer.

Numerous criticisms of the machine's play may be made. It is quite defenceless against forks, although it may be able to see certain other kinds of combination. It is of course not difficult to devise improvements of the programme so that these simple forks are foreseen. The reader may be able to think of some such improvements for himself. Since no claim is made that the above rule is particularly good, I have been content to leave this flaw without remedy; clearly a line has to be drawn between the flaws which one will attempt to eliminate and those which must be accepted as a risk. Another criticism is that the scheme proposed, although reasonable in the middle game, is futile in the end game. The change-over from the middle game to the end-game is usually sufficiently clear-cut for it to be possible to have an entirely different system for the end-game. This should of course include quite definite programmes for the standard situations, such as mate with rook and king, or king and pawn against king. There is no intention to discuss the end-game further here.

If I were to sum up the weakness of the above system in a few words I would describe it as a caricature of my own play. It was in fact based on an introspective analysis of my thought processes when playing, with considerable simplifications. It makes oversights which are very similar to those which I make myself, and which may in both cases be ascribed to the considerable moves being inappropriately chosen. This fact might be regarded as supporting the glib view which is often expressed, to the effect that "one cannot programme a machine to play a better game than one plays oneself." This statement should I think be compared with another of rather similar form. "No animal can swallow an animal heavier than himself." Both statements are, as far as I know, untrue. They are also both of a kind that one is easily bluffed into accepting, partly because one thinks that there ought to be some slick way of demonstrating them, and one does not like to admit that one does not see what this argument is. They are also both supported by normal experience, and need exceptional cases to falsify them. The statement about chess programming may be falsified quite simply by the speed of the machine, which might make it feasible to carry the analysis a move farther than a man could do in the same time. This effect is less than might be supposed. Although electronic computers are very fast where conventional computing is concerned, their advantage is much reduced where enumeration of cases, etc., is involved on a large scale. Take for instance the problem of counting the possible moves from a given position in chess. If the number is 30 a man might do it in 45 seconds and the machine in 1 second. The machine has still an advantage, but it is much less overwhelming than it would be for instance when calculating cosines.

In connexion with the question of the ability of a chess-machine to profit from experience, one can see that it would be quite possible to programme the machine to try out variations in its method of play (e.g. variations in piece value) and adopt the one giving the most satisfactory results. This could certainly be described as "learning," though it is not quite representative of learning as we know it. It might also be possible to programme the machine to search for new types of combination in chess. If this project produced results which were quite new, and also interesting to the programmer, who should have the credit? Compare this with the situation where a Defence Minister gives orders for research to be done to find a counter to the bow and arrow. Should the inventor of the shield have the credit, or should the Defence Minister?

The Manchester University Machine

In November, 1951, some months after this article was written (by Dr. Turing) Dr. Prinz was able to make the Manchester University machine solve a few straightforward chess problems of the "Mate-in-Two" type (see *Research*, Vol. 6 (1952), p. 261).

It is usually true to say that the best and often the only way to see how well the machine can tackle a particular type of problem is to produce a definite programme for the machine, and, in this case, in order to have something working in the shortest possible time, a few restrictions were imposed on the rules of chess as they were "explained" to the machine. For example castling was not permitted, nor were double moves by pawns, nor taking *en passant* nor the promotion of a pawn into a piece when it reached the last row; further, no distinction was made between mate and stalemate.

The programme contained a routine for the construction of the next possible move, a routine to check this move for legality, and various sequences for recording the moves and the positions obtained. All these separate subroutines were linked together by a master routine which reflected the structure of the problem as a whole and ensured that the subroutines were entered in the proper sequence.

The technique of programming was perhaps rather crude, and many refinements, increasing the speed of operation, are doubtless possible. For this reason, the results reported here can only serve as a very rough guide to the speed attainable; but they do show the need for considerable improvement in programming technique and machine performance before a successful game by a machine against a human chess player becomes a practical possibility.

The programme, as well as the initial position on the chess board, was supplied to the machine on punched tape and then transferred to the magnetic store of the machine.

A initial routine (sub-programme) was transferred to the electronic store, and the machine started its computation. The programme was so organized that every first move by white was printed out; after the key move had been reached the machine printed: "MATE."

The main result of the experiment was that the machine is disappointingly slow when playing chess—in contrast to the extreme superiority over human computers where purely mathematical problems are concerned. For the simple example given in the position reproduced here, 15 minutes were needed to print the solution. A detailed analysis shows that the machine tried about 450 possible moves (of which about 100 were illegal) in the course of the game; this means about two seconds per move on the average.

A considerable portion of this time had to be used for a test for self-check (i.e. after a player had made a move, to find out whether his own King was left in check). This was done by first examining all squares connected to the King's square by a Knight's move, to see (*a*) whether they were on the board at all, (*b*) whether they were empty or occupied, (*c*) if occupied, by a piece of which colour and (*d*) if occupied by a piece of opposite colour, whether or not this piece was a Knight. A similar test had to be carried out for any other piece that might have put the King in check. This test involves several hundreds of operations and, at a machine speed of 1 msec per operation, might take an appreciable fraction of a second.

The next important time-consuming factor was the magnetic transfers, i.e. the transfers of sub-programmes and data (relating to positions and moves) between the magnetic and the electronic store. It is here that improved programming technique may save time by better utilization of the electronic store, thus reducing the number of transfers (nine for every legal move in the present programme).

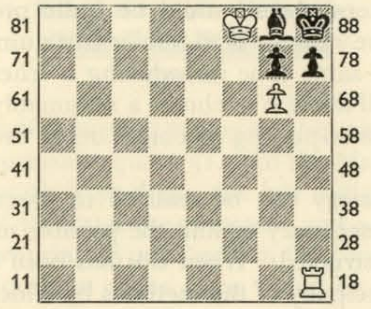
Compared with these two items, the time spent in computing the moves appeared to be of minor importance although the machine not only computed the possible moves but also the impossible, but "thinkable" moves—meaning those which either carry the piece off

the board, or lead to a collision with a piece of the same colour already on the square. These moves, however, were quickly rejected by the machine and did not contribute greatly to the total computation time.

It appears that if this crude method of programming were the only one available it would be quite impractical for any machine to compete on reasonable terms with a competent human being.

Before we conclude too easily that no computer will ever compete in a Masters' Tournament let us remind ourselves that the Manchester machine solved a problem after a few weeks tuition, which represents quite reasonable progress for a beginner.

The First Chess Problem Solved by a Computing Machine. The task set the Manchester machine was to find a move by white that would lead to a mate in the next move, whatever black might answer. The move is R—R6.



For solution of the problem by the machine the squares of the board were numbered in rather unusual fashion. The bottom row was numbered 11 to 18 (from left to right), the next 21 to 28, and so on to the top row, which was 81–88. Square 68 was thus the square in row 6, column 8. The machine has printed out all the moves which white tried out to find a solution, and has printed "MATE" after finding and recording the key move, which appears in the form "Rook to 68."

The list of moves is—

Pawn to 78.	Rook to 11.
Rook to 17.	Rook to 28.
Rook to 16.	Rook to 38.
Rook to 15.	Rook to 48.
Rook to 14.	Rook to 58.
Rook to 13.	Rook to 68.
Rook to 12.	MATE.

DRAUGHTS

The game of draughts occupies an intermediate position between the extremely complex games such as chess, and the relatively simple games such as nim or noughts-and-crosses for which a complete mathematical theory exists. This fact makes it a rather suitable subject for experiments in mechanical game playing, for although there is no complete theory of the game available, so that the machine has to look ahead to find the moves, the moves themselves are rather simple and relatively few in number.

Various forms of strategy have been suggested for constructing an automatic chess player; the purpose of such plans is to reduce the time taken by the machine to choose its move. As Prinz has shown, the time taken by any machine which considers all the possible moves for four or five steps ahead would be quite prohibitive, and the principal aim of the strategy is to reduce this number very considerably, while at the same time introducing a scheme of valuing the positions which will allow it to choose a reasonably good move. The chief interest in games-playing machines lies in the development of a suitable strategy.

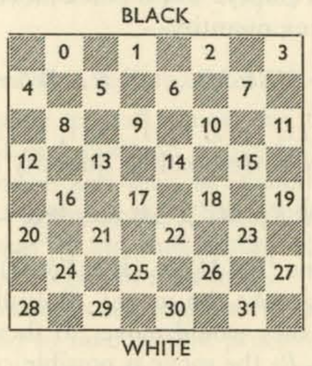
Before any strategy can be realized in practice, however, the basic programme necessary to find the possible moves and to make them must be constructed. When this has been done the strategy, which consists principally of the methods by which positions can be valued, can be added to make the complete game player. It is obviously possible to make experiments with different strategies using the same basic move-finding-and-making routine.

The basic programme for draughts, which is described in outline in the following paragraphs, is very much simpler than the corresponding one for chess. It has in fact proved possible to put both it and the necessary position storage in the electronic store of the Manchester machine at the same time. This removes the need for magnetic transfers during the operation of the programme, and this fact, together with the simplicity of the moves, has reduced the time taken to consider a single move to about one tenth of a second.

Basic Programme for Draughts

We must first consider the representation of a position in the machine. The 32 squares used in a draughts board are numbered as shown in the diagram.

A position is represented by 3 thirty-two-digit binary numbers (or words) B, W and K which give the positions of the black men (and kings), the white men (and kings) and the kings (of either colour) respectively. The digits of these words each represent a square on the board; the square n being represented by the digit 2^n.



Thus the least significant digit represents square 0 and the most significant digit represents square 31. (In the Manchester machine, where the word length is 40 digits, the last 8 digits are irrelevant). A unit in the word indicates the presence, and a zero indicates the absence of the appropriate type of man in the corresponding square. Thus the opening position of the game would be represented by*—

$$B = 1111,\ 1111,\ 1111,\ 0000,\ 0000,\ 0000,\ 0000,\ 0000$$
$$W = 0000,\ 0000,\ 0000,\ 0000,\ 0000,\ 1111,\ 1111,\ 1111$$
$$K = 0000,\ 0000,\ 0000,\ 0000,\ 0000,\ 0000,\ 0000,\ 0000$$

The positions of the white kings are indicated by the word $W\&K$, while the empty squares are indicated by the word $\sim W \& \sim B$.†

It will be seen that there are at most four possible types of non-capture moves from any square on the board. For example, from square 14 the possible moves are to squares 9, 10, 17 or 18. The machine considers all these moves in turn, but it will be sufficient to indicate here the way in which it deals with one of them—say the move 14–18.

* All binary numbers are written in the convention used for the Manchester machine, i.e. with their *least* significant digit on the left.

† $W\&K$ stands for the logical product of W and K (sometimes also known as the result of collating W and K). $\sim W$ stands for the negation of W, i.e. the word obtained by writing 1's for 0's in W, and vice versa (see Chapter 15).

This type of move, which consists of adding 4 to the number of the square, corresponds to multiplying the appropriate digit in the position word by 2^4. A move of this type can be made by any black man, but only by a white king; it cannot be made from squares 28, 29, 30 or 31 nor can it be made unless the square to which the man is to be moved is empty. For a black move, the machine therefore forms the following quantity—

$$Y = \{(B \& M) \times 2^4\} \& {\sim} W \& {\sim} B$$

where $M =$ 1111, 1111, 1111, 1111, 1111, 1111, 1111, 0000
For a white move, the corresponding quantity is—

$$\{(W \& K \& M) \times 2^4\} \& {\sim} W \& {\sim} B$$

In these expressions $(B \& M)$ or $(W \& K \& M)$ give all the men on the board who could make the move; multiplying this by 2^4 give the squares to which they would move. If these squares are empty (collate with ${\sim} W \& {\sim} B$) the move is possible.

The quantity Y thus represents all the possible moves of this type. To consider a single one of these, the largest non-zero digit of Y is taken and removed from Y. The word consisting of this single digit known as θ, gives the square to which the man is moved. The quantity $\phi = \theta \times 2^{-4}$ is then formed and gives the square from which the man was moved. For a black move, the quantity—

$$B' = B \not\equiv \theta \not\equiv \phi$$

will then give the new position of the black men. If $K \& \phi$ is not zero, the man moved was a king so that $K' = K \not\equiv \theta \not\equiv \phi$ gives the new position of the kings. If $K \& \phi$ is zero, the man moved was not a king. The new position of the kings will therefore be unaltered unless the man has kinged during this move—in other words unless $\theta > 2^{28}$ in which case $K' = K \not\equiv \theta$.

Relatively simple modifications of this scheme are needed to deal with white moves and non-capture moves of other types. Capture moves are somewhat more complicated as multiple captures must be allowed for. Furthermore, all the possible captures must be made or the machine will render itself liable to be huffed. This leads to a considerable complication which it is not possible to describe fully here, but the basic scheme is not altered.

The machine considers all the possible moves of one type before starting the next, so that in order to describe a position fully, it is necessary to store the word Y, which indicates the moves still to be considered, as well as the position words B, W and K. It is also necessary to keep a record of the type of move being considered. This is done with the aid of a further parameter word P which also contains the value associated with the position. The whole storage required for a position is thus reduced to the 5 thirty-two-digit words B, W, K, Y, and P.

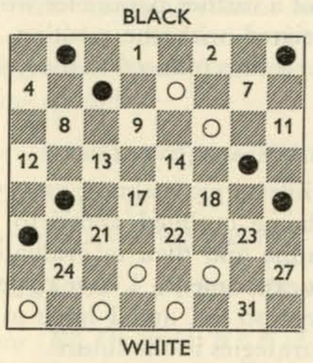
Valuation of Positions and Strategy

It should be possible to graft almost any type of strategy on to the move-finding scheme outlined above to produce a complete draughts-playing routine and then to evaluate the effectiveness of the strategy by direct experiment. I have done this with two rather simple types of strategy so far, and I hope to be able to try some rather more refined strategies in the future.

For demonstration purposes, and also to ensure that a record of the game is kept, and to take certain precautions against machine error, the move-finding sequence and the associated strategy have been combined with a general game-playing routine which accepts the opponent's moves, displays the positions, prints the move, and generally organizes the sequence of operations in the game. It is rather typical of logical programmes that this organizing routine is in fact longer than the game-playing routine proper. As its operations, though rather spectacular, are of only trivial theoretical interest, I shall not describe them here.

The first, and simplest, strategy to try is the direct one of allowing the machine to consider all the possible moves ahead on both sides for a specified number of stages. It then makes its choice, valuing the final resulting positions only in terms of the material left on the board and ignoring any positional advantage. There is an upper limit to the number of stages ahead that can be considered owing to limitations of storage space—actually six moves, three on each side, are all that can be allowed. In practice, however, time considerations provide a more severe limitation. There are on an average about ten possible legal moves at each stage of the game, so that consideration of one further stage multiplies the time for making the move by a factor of about ten. The machine considers moves at the rate of about ten a second, so that looking three moves ahead (two of its own and one of its opponents), which takes between one and two minutes, represents about the limit which can be allowed from the point of view of time.

This is not sufficient to allow the machine to play well, though it can play fairly sensibly for most of the game. One wholly unexpected difficulty appears. Consider the position on the following board.



In this position, the machine is aware that its opponent is going to king next move. Now a king is more valuable than a man—the actual values used were three for a king and one for a man—so that if the opponent kings the machine effectively loses two points. The only way it can stop this is by offering a man for sacrifice, because then, by the rules of the game, the sacrifice must be taken at once. If the machine does this, it will lose only one point, and as it is not looking far enough ahead, it cannot see that it has not prevented its opponent from kinging but only postponed the evil day. At its next move it is still faced with the same difficulty, which it tries to solve in the same way, so that it will make every possible sacrifice of a single man before it accepts as inevitable the creation of an opponent's king. In fact, when faced with this position, the machine played 19—23, followed by 16—21 and 20—24.

This, of course, is a fatal flaw in the strategy—and not one it would have been easy to discover without actually trying it out. An opponent who detected this behaviour—and it is extremely conspicuous in play—would only have to leave his man on the point of kinging indefinitely. The machine would then sacrifice all its remaining men as soon as the opportunity offered.

In order to avoid this difficulty, the second strategy was devised. In this the machine continues to investigate the moves ahead until it has found two consecutive moves without captures. This means that it will be able to recognize the futility of its sacrifice to prevent kinging. It is still necessary to impose an over-riding limit on the

number of stages it can consider, and once more, considerations of time limit this. However, as no move is continued for more than two stages unless it leads to a capture, it is possible to allow the machine to consider up to four stages ahead without it becoming intolerably slow. This would mean that it would consider the sacrifice of two men to be of equal value to the creation of an

	Machine	*Strachey*
1.	11—15	23—18
2.	7—11	21—17
3.	8—12	20—16[1]
4.	12—21(16)	25—16(21)
5.	9—14![2]	18—9(14)
6.	6—20(16,9)[3]	27—23
7.	2—7[4]	23—18
8.	5—8	18—14
9.	8—13[5]	17—8(13)
10.	4—13(8)	14—9
11.	1—5[6]	9—6
12.	15—19	6—1 K
13.	5—9	1—6?[7]
14.	0—5![8]	6—15(10)
15.	11—25(22,15)	30—21(25)
16.	13—17	21—14(17)
17.	9—18(14)	24—21
18.	18—23	26—22
19.	23—27	22—17
20.	5—8[9]	17—14
21.	8—13	14—9
22.	19—23	9—6
23.	23—26[10]	31—22(26)
24.	27—31 K	6—2 K
25.	7—10	2—7
26.	10—15	21—16?[11]
27.	3—10(7)	16—9(13)
28.	10—14	9—6
29.	15—19	6—2 K
30.	31—27[12]	2—6
31.	27—31[12]	6—10
32.	31—26[13]	10—17(14)
33.	19—23	29—25
34.	26—31[14]	

Notes—

1. An experiment on my part—the only deliberate offer I made. I thought, wrongly, that it was quite safe.
2. Not foreseen by me.
3. Better than 5—21(9,17).
4. A random move (zero value). Shows the lack of a constructive plan.
5. Another random move of zero value, actually rather good.
6. Bad. Ultimately allows me to make a King. 10—14 would have been better.
7. A bad slip on my part.
8. Taking full advantage of my slip.
9. Bad. Unblocks the way to a King.
10. Sacrifice in order to get a King (not to stop me kinging). A good move, but not possible before 19—23 had been made by chance.
11. Another bad slip on my part.
12. Purposeless. The strategy is failing badly in the end game.
13. Too late.
14. Futile. The game was stopped at this point as the outcome was obvious.

opponent's king, and as there is a random choice between moves of equal value, it might still make this useless sacrifice. This has been prevented by reducing the value of a king from 3 to $2\frac{7}{8}$.

With this modified strategy, the machine can play quite a tolerable game until it reaches the end game. It has always seemed probable that a wholly different strategy will be necessary for end games. The game given on page 303, which is the first ever played using the strategy, brings this point out very clearly.

NIM

A considerably easier game which the machine can be programmed to play is the one known as nim. Probably a variation of this was known to the Chinese—certainly in its present form many people have met it. We have chosen to deal with this comparatively trivial game in detail because of its topical interest. Thousands of people will have seen *Nimrod*, the computer built by Ferranti Ltd. for the Science Exhibition of the Festival of Britain. This special-purpose machine was designed to show the main features of large electronic digital computers, and the game of nim was chosen as an interesting but simple demonstration problem. The game itself is as follows—

Initially we have any number of heaps, each containing any number of tokens (usually matches). In the simplest form, two contestants play alternately, and may pick up as many matches as they wish at one time from *one* pile, but they must take at least one match. The aim is to avoid taking the last match of all—or there is another variation where the aim is to take the last match or group of matches.

The so-called *multiple game* differs from this only in that the number of heaps altered in any move may take any value from one up to a pre-assigned maximum k. Of course, to prevent complete triviality, k must be less than N, the total number of heaps.

The detailed theory of nim was worked out long ago and, apart from the initial distribution of the matches, no element of chance need enter into the game. This theory is very simple, but it becomes clearer for the non-mathematician if we use the concept of a binary number, introduced elsewhere (see page 33).

We can now proceed to give a working rule for the game of nim. We would like to find a *winning position* having the following characteristics—

(*a*) It is impossible, when faced by a winning position, to make a move which will leave a winning position.

(*b*) Faced with any other than a winning position, it is possible to make a move resulting in a winning position.

(*c*) If at any stage of the game a player A can convert a position into a winning position, it is possible for A to win, and impossible for his opponent B to do so unless A makes a mistake. A wins by leaving a winning position at every succeeding move on his part.

Such winning positions can be achieved and are recognized as follows: For any given configuration, express the number of matches in each heap as a binary number. Suppose, for example, that we have four heaps, A, B, C and D, containing respectively 7, 4, 3 and 2 matches. These are represented—

	4	2	1	
A	1	1	1	(7)
B	1	0	0	(4)
C	0	1	1	(3)
D	0	1	0	(2)

We write these down as above, one under the other, and add up each column, e.g., in the above example, we get

	4	2	1
Sum:	2	3	2

Now the "secret" of a winning position is that every column should be divisible by $k + 1$; k being the maximum number of heaps which can be altered in any one move. Thus the example quoted above cannot represent a winning position whatever our initial choice of k. However, suppose we have $k = 1$; then consider the position—

	4	2	1	
A	1	0	1	(5)
B	1	1	1	(7)
C	0	1	1	(3)
D	0	0	1	(1)
Sum:	2	2	4	

This is a winning position, but would not be so if we had previously fixed $k = 3$, for example.

To convert an "unsafe" into a winning position, we deal with a column at a time. Consider our previous example with $k = 1$.

	4	2	1	
A	1	1	1	(7)
B	1	0	0	(4)
C	0	1	1	(3)
D	0	1	0	(2)
Sum:	2	3	2	

We start with the "most-significant," or left-hand column. This sum is divisible by $k + 1$, so we proceed to consideration of the next column. The sum here is 3, which is not divisible by $k + 1$, so we choose any heap, say *D*, having a one in this column. We remove this 1 (which is equivalent to subtracting 2 from *D*), and put 1 in every less-significant (or right-hand) column of this heap (which in this case is equivalent to adding 1, though if we had chosen to modify *A* instead, it would have meant no change in the last column). That is, we make the minimum move which removes the 1 in the "unsafe" column. Thus we remove 1 from *D*, and so alter its binary representation to 001.

Now our representation is—

	4	2	1	
A	1	1	1	(7)
B	1	0	0	(4)
C	0	1	1	(3)
D	0	0	1	(1)
Sum:	2	2	3	

and we see that we have made the sum of column 2 divisible by $k + 1$ at the expense of column 1. However, we shall now proceed to adjust column 1. To avoid altering more than k heaps in one move, we must alter one or more of the heaps already affected if,

by so doing, we can achieve the desired result, rather than select a fresh heap.

Now, in this case, we wish to remove 1 from column 1 of some heap. Since heap *D* has already been altered, we choose this—it has a 1 in this column.

So, at the end of our move, we have removed two matches from heap *D*, and leave the winning position—

	4	2	1	
A	1	1	1	(7)
B	1	0	0	(4)
C	0	1	1	(3)
D	0	0	0	(0)
Sum:	2	2	2	

In adapting this game for the universal computer, we allow a maximum of eight heaps, with not more than thirty-one matches in a heap. In Nimrod the more stringent restrictions to four heaps, each with a maximum content of seven matches, were applied to simplify the problems of demonstration.

Possible positions with which the machine may be faced are as follows—

(*a*) At least $k + 1$ heaps contain more than one match.

(*b*) The number of heaps containing more than one match lies between 1 and k (inclusive).

(*c*) No heap contains more than one match. Not all heaps are empty.

(*d*) All heaps are empty.

In case (*a*), we follow the so-called *normal routine*, which aims at leaving column sums all divisible by $(k + 1)$.

In case (*b*), we want to leave $r\ (k + 1) + 1$ heaps containing one match, and no heaps with more than one, where r may have any non-negative integral value (i.e. $r = 0, 1, 2, \ldots$).

In case (*c*) the same applies. If only one heap is left, containing one match, we have no choice of move, but this need not be treated separately.

In case (*d*), the game is over. Special investigation has to be used to detect this case. In all other cases, if the normal routine cannot succeed in its purpose, i.e. if the machine is faced with a winning position—a random move can, and must, be made. But, in this situation, this obviously cannot be done.

Thus the routine breaks up naturally into the following parts—

(i) Entry
(ii) Determination of case
(iii) Normal Routine
(iv) Cases (*b*) and (*c*)
(v) Treatment of zero case (*d*)
(vi) Random move
(vii) Emergence.

There is no need to give further details of the programme, but an example is given of how the machine would tackle a specific game.

Suppose initially that we have four heaps, containing respectively 7, 4, 5 and 2 matches; that $k = 2$; and that the machine moves first.

(i) Entry—

	4	2	1	
A	1	1	1	(7)
B	1	0	0	(4)
C	1	0	1	(5)
D	0	1	0	(2)

(ii) Determination of case—

There are 4 non-zero, non-unit heaps, so we are dealing with case (*a*).

(iii) Normal routine—

	4	2	1	
A	1	1	1	(7)
B	1	0	0	(4)
C	1	0	1	(5)
D	0	1	0	(2)
Sum:	3	2	2	

The sum of column 4 is divisible by $k + 1$ so we need not modify it.

The sum of column 2 is 2, and is not divisible by $k + 1$, so we need to modify any heap having a 1 in this column—say heap *A*.

According to the rules, we then get—

	4	2	1	
A	1	0	1	(5)
B	1	0	0	(4)
C	1	0	1	(5)
D	0	1	0	(2)
Sum:	3	1	2	

And we note that heap *A* has been modified, and should be again modified whenever possible. Sum of column 2 is still not divisible by $k + 1$, so this time we modify heap *D* to obtain—

	4	2	1	
A	1	0	1	(5)
B	1	0	0	(4)
C	1	0	1	(5)
D	0	0	1	(1)
Sum:	3	0	3	

Column 2 is now divisible by $k + 1$ and, proceeding to the next column, we see this condition is also satisfied here, so the move has been completed and a winning-position left, the means to this end being the removal of two matches from *A*, and one from *D*, leaving 5, 4, 5 and 1. (If column 1 had needed adjustment, we should have had to modify one or both of heaps *A* and *D*, since these had already been affected.)

Suppose the opponent now makes a move leaving 0, 4, 2 and 1 as the contents of the respective heaps. It is now for the machine to move again.

(i) Entry—

	4	2	1	
A	0	0	1	(0)
B	1	0	0	(4)
C	0	1	0	(2)
D	0	0	1	(1)

(ii) Determination of case.

There are 3 non-zero, non-unit heaps, so we are dealing with case (*b*). Thus we want to leave 1, or 4, or 7 . . . unit-heaps. Clearly we can only leave 1 unit heap in this case.

(iv) Cases (*b*) and (*c*).

We remove all matches from heaps *B* and *D*, which affects only *k* heaps, and leaves just one unit heap as required.

The opponent is now forced to remove the last match, and the machine wins the game.

Chapter 26

THOUGHT AND MACHINE PROCESSES

Cogito, ergo sum—DESCARTES
I do not think, therefore I am not—DR. STRABISMUS (whom God preserve) of Utrecht. President of the Anti-cartesian Society

SO FAR WE HAVE DESCRIBED the construction of these new digital computers, and tried to show how useful they can become by doing routine computations. If we are to complete the story we must also try to assess the limitations of the machines which we can build today, and, if possible, to discuss any limits to the performance of machines which may be built in the future. We shall try to compare the processes which go on inside them with those which are responsible for the thoughts in our own minds. This subject is far too complicated to be dealt with in a single chapter, but we shall try to describe some of its more important aspects. We shall begin by giving an account of some of the astonishing feats of mental arithmetic which are demonstrated by those rare individuals who are known as "calculating prodigies." Many accounts of these men have appeared, of which one of the best known is to be found in W. W. R. Ball's book,* to which the reader is referred for a comprehensive historical account of the subject.

At rare intervals there have appeared men and boys who display extraordinary powers of mental arithmetic. In a few seconds they can give the answer to questions which an expert mathematician could obtain only in a much longer time with the aid of pencil and paper. Some of them have remained otherwise illiterate; others, such as Gauss, Ampère, and Bidder, have risen to positions of eminence as mathematicians, physicists, or engineers. Many of them seem to have taught themselves the rules of arithmetic in their childhood, and to have learnt the multiplication table by playing with pebbles. Few of these prodigies have been able to explain in detail how they achieve their apparently miraculous results, but two of the most remarkable of them have been kind enough to discuss their methods with us. We are therefore much indebted to Dr. A. C. Aitken, F.R.S., Professor of Mathematics in Edinburgh

* *Mathematical Recreations and Essays*. See also *Common Sense and Its Cultivation*, by Hanbury Hankin, and *Mental Prodigies*, by Fred Barlow.